Atty. Docket No. MS306137.01

HOSTED APPLICATION AS A DESIGNER IN AN INTEGRATED DEVELOPMENT ENVIRONMENT

by

Svetlana Loksh, Thomas E. Quinn and Reza Chitsaz

MAIL CERTIFICATION

I hereby certify that the attached patent application (along with any other paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on this date **February 13, 2004**, in an envelope as "Express Mail Post Office to Addressee" Mailing Label Number **EV373131504US** addressed to the Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

_____

Himanshu S. Amin

Title: HOSTED APPLICATION AS A DESIGNER IN AN INTEGRATED
DEVELOPMENT ENVIRONMENT

5

TECHNICAL FIELD

This invention is related to a software development environment, and more
specifically to architecture that facilitates bringing an unmanaged application into the
environment as a designer.

10

BACKGROUND OF THE INVENTION

Software integrated development environments (IDEs) heretofore are not capable
of integrating the functionality of an unmanaged hosted application into the development
workspace. Thus, writing rich designers is normally a very complicated and time-

15   consuming task that must consider very tight integration with the hosted applications.

What is needed is architecture that integrates the unmanaged application into the
design environment in a coherent yet usable manner such that the developer can use the
hosted application's native functionality yet leverage rich productivity features provided
by the IDE.

20

SUMMARY OF THE INVENTION

The following presents a simplified summary of the invention in order to provide
a basic understanding of some aspects of the invention. This summary is not an extensive
overview of the invention. It is not intended to identify key/critical elements of the

25   invention or to delineate the scope of the invention. Its sole purpose is to present some
concepts of the invention in a simplified form as a prelude to the more detailed
description that is presented later.

The present invention disclosed and claimed herein, in one aspect thereof,
comprises architecture that integrates the unmanaged application as a hosted application

30   into the design environment in a coherent yet usable manner such that the developer can
use the hosted application's native functionality yet leverage rich productivity features
provided by the IDE. In support thereof, there is provided an IDE having a hosting

component that comprises an application adaptor that provides the interface from the unmanaged application to the IDE. The IDE includes a designer framework that provides user access to designer capabilities of the IDE. The host adaptor interfaces to the designer framework either directly and/or through an integration interface. Thus, the

5   developer/user can then access the native functionality of the unmanaged application as well as the IDE functionality. This provides rich integration between the development environment and the hosted application, the result of which is the designer interacting with the IDE.

To the accomplishment of the foregoing and related ends, certain illustrative

10   aspects of the invention are described herein in connection with the following description and the annexed drawings. These aspects are indicative, however, of but a few of the various ways in which the principles of the invention may be employed and the present invention is intended to include all such aspects and their equivalents. Other advantages and novel features of the invention may become apparent from the following detailed

15   description of the invention when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a system of the present invention.

FIG. 2 illustrates a flow chart of a process for integrating native application

20   functionality in an IDE.

FIG. 3 illustrates a flow chart of a process for OLE document development in accordance with the present invention.

FIG. 4 is a sample window showing integrated functionally in accordance with the present invention.

25   FIG. 5 illustrates a window showing an integrated and cascading menu feature of the present invention.

FIG. 6 illustrates a block diagram of a computer operable to execute the disclosed architecture.

FIG. 7 illustrates a schematic block diagram of an exemplary computing

30   environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order

5   to provide a thorough understanding of the present invention. It may be evident, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the present invention.

As used in this application, the terms "component" and "system" are intended to

10   refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a server and the server can be a component. One or more

15   components may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers.

Referring now to FIG. 1, there is illustrated a system 100 of the present invention. The system 100 is an enhanced Integrated Development Environment (IDE) 102 that facilitates bringing the native functionality of an unmanaged application 104 into the

20   development environment 102. In furtherance thereof, the IDE 102 includes a hosting component 106 that interfaces to the unmanaged application 104 at least by way of a host adaptor 108. The host adaptor 108 is part of the hosting component 102 and is specific to the particular unmanaged application 104. That is, where the unmanaged application 104 is a word processor, the host adaptor provides interface capabilities specific to the word

25   processor application. Similarly, a spreadsheet application would have its own specific host adaptor to facilitate bringing its native functionality into the IDE 102.

The hosting component 106 also includes a designer framework 110 that interfaces to the host adaptor 108, and functions as an interface between the hosting component 106 and the IDE 102. The host adaptor 108 also interfaces to the designer

30   framework through an integration interface 112, which functions as an interface for third

party applications whose native functionality can also be brought into the IDE 102, provided there is a host adaptor 108 specific to that third party application.

The hosting component 106 also includes a document hosting subcomponent 114 that interfaces to the designer framework 110 and the unmanaged application 104 to

5   facilitate hosting an associated document of the application 104.

As indicated previously, the system 100 is capable of integrating the native functionality of a plurality of unmanaged applications (denoted as Unmanaged App$_1$...Unmanaged App$_N$). Thus, the IDE 102 includes a corresponding plurality of hosting components (denoted Hosting Component$_1$...Hosting Component$_X$). Each

10   hosting component includes the host adaptor, designer framework, integration interface and document hosting subcomponent. Note that although these components are shown as separate entities, these entities can be combined in any manner deemed suitable to bring the native functionality of the unmanaged application into the IDE.

The system 100 is particularly suited for an OLE (Object Linking and

15   Embedding) technology where the IDE supports OLE, and the document hosted by the document hosting subcomponent 114 is an OLE document. Thus, any application supporting OLE document interfaces can be integrated into the IDE 102. However, the present invention is not limited to such an OLE technology, and contemplates any other suitable technology for use in the IDE.

20   The innovation behind hosting document-based rich applications (*e.g.*, a word processor and spreadsheet application) inside the IDE is that these applications are treated as visual designers. Hosted applications are fully functional, but take advantage of the rich functionality provided by the IDE. When hosted in the IDE, host documents can be visually designed *via* interaction with the different tool windows. For example, a

25   Toolbox can be used to drag and drop controls and components onto the designer. Controls on the documents can be selected in the designer and their properties edited in a Property window. Another operation implements data binding functionality in the application *via* a Data Source Picker Tool window, for example.

The present invention turns a hosted application into a visual designer without

30   diminishing any of the native functionality provided by the hosted application. This effectively turns the hosted application into a design surface that developers can drag and

drop objects onto in order to rapidly create solutions and applications. Moreover, the disclosed approach provides a familiar and consistent development model developing applications in the IDE. Additionally, the enhanced IDE of the present invention keeps the number of concepts to a minimum while also minimizing the number of design

5   paradigms.

Referring now to FIG. 2, there is illustrated a flow chart of a process for integrating native application functionality in an IDE. While, for purposes of simplicity of explanation, the one or more methodologies shown herein, *e.g.*, in the form of a flow chart, are shown and described as a series of acts, it is to be understood and appreciated

10   that the present invention is not limited by the order of acts, as some acts may, in accordance with the present invention, occur in a different order and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all

15   illustrated acts may be required to implement a methodology in accordance with the present invention.

At 200, an IDE is provided. At 202, the hosting component is provided in the IDE to facilitate hosting the unmanaged application. At 204, the designer framework so provided and interfaced to the hosting component. At 206, the host adaptor specific to

20   the unmanaged application is provided as part of the hosting component. At 208, the host adaptor is interfaced directly to the designer framework and indirectly through the integration interface for third party applications. At 210, the document hosting subcomponent is provided and interfaced to the designer framework and the unmanaged application. At 212, the hosted document can now be manipulated using both the

25   designer functionality and native functionality of the unmanaged application. The process then reaches a Stop block.

Referring now to FIG. 3, there is illustrated a flow chart of a process for OLE document development in accordance with the present invention. In this particular process implementation, the IDE environment supports OLE technology. At 300, the

30   developer opens the IDE environment. At 302, the unmanaged application is selected. At 304, a new document is opened or an existing document is opened for editing. At 306,

security settings can be set for the document to, for example, set various levels of permissions or restrictions according to where the document will be processed. At 308, the developer/user manipulates the document and/or document content using capabilities of one or both of the designer framework and native functionality of the hosted

5   application. The process then reaches a Stop block.

HOST ADAPTORS

When the hosted application is opened as a designer, the hosted menus will merge with the menus of the IDE. Certain menu items will be enabled and others will be

10   disabled depending on whether the hosted application has focus or not. That is, if the IDE has focus, certain menu item will be disabled within the hosted application. Contrariwise, if the hosted application has focus, certain menu items of the IDE will be disabled. This is but only one function of the host adaptor.

Referring now to FIG. 4, there is a sample window 400 showing integrated

15   functionally in accordance with the present invention. The window 400 is representative of a hosted spreadsheet application (*e.g.*, related to an EXCEL-brand spreadsheet application). The window 400 shows host menus 402 merged with the IDE menus 404 of the IDE, a host context menu 406 with an additional View Code command 408, a Solution Explorer 410 showing a project structure 412, a Property window 414 showing

20   properties 416 of the selected spreadsheet, and a Toolbox 418 with a list of controls and components 420 that can be added to a hosted document 422.

The Property window displays properties for code-behind project items and other designer project items. The hosted application pushes its properties to the Property window. The hosted application can choose to propagate properties to the Property

25   window. The hosted application provides a list of properties (*e.g.*, property name and property type).

Any documents that contain embedded or linked objects can be opened in the IDE. Selecting an embedded or linked object causes the menus and toolbars to change for the selected object.

30   If the first radio button in the Tools options is selected, the adaptor captures all accelerators and routes the accelerators to a table of shortcuts (or key mappings) in the

IDE that are dynamically created. Anything not handled is passed down by the IDE table back to the hosted application for handling.

Right clicking on the designer will bring up the standard spreadsheet context menu. The additional menu command "View Code" is the first command in the context

5   menu. Having the "View Code" command at the top of the list provides consistency with the context menus of other programs. Selecting the View Code command off a spreadsheet cell navigates the developer to the code editor for that worksheet. The cursor will be set to wherever it was set to before. By default, this means the cursor will be set to the top of the editor. Right clicking on a View Control will display the context menu

10   for that View Control. Selecting the View Code command off a View Control navigates the developer to a code editor of the IDE and sets the cursor to wherever it was set to before.

Right clicking on a worksheet tab brings up a context menu of the spreadsheet application. This context menu contains a View Code command that if selected navigates

15   the user to code behind the file for that spreadsheet in the IDE code editor, when selected. The cursor placing behavior in the code editor is the same with this View Code command as it is with the View Code command on the designer.

Referring now to FIG. 5, there illustrated a window 500 showing an integrated and cascading menu feature of the present invention. Integration of the hosted

20   application (also known as a server application) with the IDE can result in conflicts between similarly named menu items. Here, integration of the IDE and the hosted application causes a menu conflict such that at least the View menu items are in conflict. The IDE modifies its existing View menu 502 to include the hosted application menu 504 (accessible through and notated as Server View Commands). The IDE presents to the

25   user text that clearly identifies the location of the server View menu options within the IDE View menu 502. An expansion arrow associated with the Server View Commands menu option facilitates accessing the cascaded server submenu items.

EXAMPLE SCENARIO

30   In the IDE, the user creates a new spreadsheet project. Spreadsheet-related project items are created in the Solution Explorer and the spreadsheet workbook, and

sheets are displayed in the document window, as a designer. The spreadsheet document is designed inside the IDE. The disclosed architecture merges the menus of the IDE and hosted application, and toolbars are available within the spreadsheet application of the designer, providing productivity features and the IDE features all in one environment.

5   The spreadsheet can be edited and modified using the full fidelity of the spreadsheet application, as well as use the IDE tool windows and editors to customize the spreadsheet using managed code. From the Toolbox, the user can drag and drop a managed button onto the document. The button can be selected and its properties modified using a property grid. Customizations can be made inside an automatically generated event

10   handler by double clicking on the button and entering Code View.

DESIGNER MODES

The hosted application operates in a Design mode and a Runtime mode to distinguish between design actions and runtime actions within the hosted application.

15   The set of actions the user can perform in each mode may differ from application to application, and is dependent on the hosted application regardless of whether or not in the IDE. In one implementation, the hosted application is in design time mode, and stays in design time mode when hosted in the IDE. This experience facilitates dragging controls onto the designer, selecting the control to view and edit its properties, and double clicking

20   on the control to enter Code View. The host adaptor for an unmanaged application pushes and keeps the document in design mode when in the IDE.

Another mode is a "run" mode. When the application is running or is being debugged, it is in a "run" mode. In this mode, only native application functionality is available. Once debugging begins, the code editor becomes the active designer, and all

25   accelerators are handled by the IDE.

DOCUMENT VIEWS

A document that is hosted in the IDE can have at least two views: a Design view, and a Code View. Generally, Design view is the design/normal view of the application.

30   Design view is enabled for documents that support OLE Document technology. The user experience when working on these documents is substantially similar to the experience of

using the original hosted application. This means that a developer can design templates/documents, create and modify documents, and use special host features such as formatting, charting, and schema mapping, for example. This behavior is achieved by leveraging OLE technologies and hosting the hosted application in the IDE.

5   The Design view is capable of displaying new and existing documents in formats supported by the hosted application. For instance, the designer will handle documents in binary and in XML format. This is achieved by leveraging OLE Document technologies to fully integrate the document into the IDE to enable operations such as the drag and drop of controls onto the designer, selection context, and double clicking on controls to

10   write code. The designer is capable of displaying the entire document content, including but not limited to formatting, styles, schema maps, and embedded objects.

When in the IDE the hosted application is in "design mode". The application hosted in the IDE is fully functional, *i.e.*, the hosted application's menus, commands, context menus, and accelerators are all available to the user when developing a solution

15   in the IDE. At the same time, there are features and experiences that are additions to the host native functionality. As indicated herein, examples include interactions with the IDE tool windows, the capability to right click on the designer (document) and select the "Code View" command to navigate to the code behind file, select a function key (*e.g.*, F5) to start debugging, etc. These features significantly improve the developer

20   experience inside the IDE and aid the developer in designing applications.

There are cases when the user experience inside the hosted application can differ from the experience in the IDE. For example, if a "Tools > Macro" menu option is not available, different actions can occur when certain key combinations are exercised (*e.g.*, if Alt + F11 is selected, a Macro environment will be loaded). Specific cases where the

25   hosted application behavior differs from that of when it is outside of the IDE are associated with the host adaptor. The host adaptor processes these differences in behavior by turning these host features off or modifying their behavior. The purpose of doing this is to improve the developer experience, and increase the integration between the hosted application and the development environment.

30   Note that if an application document has multiple "subdocuments" associated with it (*e.g.*, multiple worksheets associated with a workbook), then there will be a code

view associated with each of the "subdocuments" (*i.e.*, a code-behind file associated with each one of the worksheets). Documents that are being hosted and customized in the IDE have an associated code view.

Code View exposes a code-behind file associated with a document. The code
5   window associated with the document presented by the designer can be activated in multiple ways. One way is by clicking the "View Code" button on the Solution Explorer or choosing menu item off the VS View menu "View Code" when either a designer or corresponding code project item is selected in the Solution Explorer. In the case when a designer is selected in the Solution Explorer, the code window corresponding to the
10   currently active sub-document is opened. Another way is by choosing the "View Code" menu off the designer context menu (note that the cursor is not on the mapped element). Still another way is by choosing "View Code" off a Project item context menu in the Solution Explorer.

In all actions above, the code window is loaded with the cursor set at the top of
15   the window. When code window is brought up by double clicking on a control in the designer. Code generation (*i.e.*, handler definition) and navigation then take place. In this case, the cursor is set inside the default control event handler. The handler is generated if it does not exist already. The hosted application provides sufficient data to the IDE to know which event handler to generate. If the default event handler has
20   already been generated, double clicking on a control will navigate the user to the code editor and place the cursor in the default event handler.

VIEW CONTROLS

View controls are objects that are aggregated to support data binding, and provide
25   an event model that the developer can use to respond to events fired by the view control. In order to increase the visibility and discoverability of view controls, the controls are placed in the Toolbox as "controls" that a user can drag and drop onto the application designer. The following description is about design time tools that support discovering, creating, deleting, and working with view controls in a word processing application.
30   However, it is to be appreciated that a spreadsheet application also has view controls associated therewith.

There is some level of support natively in the word processor application (and the spreadsheet application), for example, that can be used for creating and working with view controls. To improve the experience, additional tools support are added around view controls to increase the visibility and discoverability of this feature, as well as

5   improve the overall developer experience of working with them. In order to provide a model that the developer will be familiar with the experience around view controls mimics working with managed controls, *e.g.*, selection context and property window support, adding view controls from the toolbox, and double clicking on view controls to navigate to code view.

10   Features include, but are not limited to, regular drag and drop support of view controls from a Toolbox, default naming for view controls added *via* the Toolbox, selection context for view controls (*e.g.*, selecting a view control in the designer displays the properties and in the property window, and selecting a view control in the property grid dropdown selects the view control in the designer), double clicking on a view control

15   to navigate to code editor, default event handler generation, dynamic synchronization of code behind and view controls that are added to the designer, support for renaming view controls *via* a Property window, and event dropdown support for view controls in code editor. Other features can include Toolbox accessibility for view controls supporting "Enter" and "double-click" in the Toolbox to add a view control), showing Bookmarks,

20   handling of overlapping bookmarks, function key (F1) support for view control items in the Toolbox, Toolbox accessibility (supporting Ctrl+C and Ctrl+V for adding view controls, support for deleting Bookmarks, providing a user the option to demote a view control to a regular Bookmark (to reduce clutter), tabbing between view controls, and toolbox support for Shapes. A Bookmark view control is supported for the word

25   processor that functions to span from a single character to multiple paragraphs.

A developer can use a word processor XML Structure task pane to apply schema to a document when in the IDE. The mapping process creates XML node view control for each mapped element. Default naming of the XML node view control will be the same as the schema mapped element. To ensure uniqueness of the schema mapped view

30   control variable an X is appended to the name if collisions occur (where X = Integer that increments as additional collisions occur).

With respect to the spreadsheet application view controls, there is provided a Named-Range control and a ListObject control. Named-Range is a mechanism to name ranges easily and maintains a name table for each spreadsheet that ensures unique naming for each named range. ListObject is an extension is a spreadsheet range that is designed

5   to provide an enhanced data experience, including the capability to automatically grow/shrink to the dataset to which it is applied. A list objects comprises rows and columns that are exposed as ListRows and ListColumns.

COMPONENT TRAY

10   The Component Tray facilitates a way to display or visualize non-visual components when hosting the application. There are components of development that do not have visual representation at runtime, such as data adapters and associated components. The Component Tray provides a mechanism for visually recognizing and manipulating non-visual components at design time. A developer can drag and drop, edit

15   properties, and visualize these components. One of the ways to add a component to a view is by dragging a component from the toolbox and dropping it onto the document, setting their properties, delete them, etc. The non-visual component becomes a named class member instance of the class behind the document.

The tray is present if there are non-visual components on the document. If there

20   are no non-visual components, then the tray completely disappears. The user inserts the first non-visual component onto a view by dropping it onto the main portion of the view, which causes the tray to open, and the non-visual component to be moved into the tray. Subsequent non-visual components can be dropped either onto the visual form or directly onto the tray.

25   If a visual item is dragged over the component tray, a no-drop icon is displayed and the drop performs a no-op. The tray disappears when the last non-visual component is deleted from the form and selection returns to the designer. The background of the tray is the system window background color. The components are auto-arranged to match the current width of the tray, so horizontal scrolling is never needed. A vertical scrollbar will

30   appear in the tray, if necessary. A vertical scroll bar is added to the tray if more components are added than can be displayed in the tray frame. A horizontal scroll bar is

added to the tray if a component is placed so that it appears past the component tray's left or right boundaries. Changes to the component tray do not dirty the document.

Reviewing some of the features of the disclosed architecture, the IDE Toolbox window enables drag and drop or the drawing of managed controls onto a designer

5   surface. A Data Source window enables drag and drop of managed data components and controls onto a designer surface. The IDE Property window enables the manipulation of properties of the document and components on it. The IDE Component Tray window hosts non-visual elements dropped on documents such as data adaptors and data connectors, for example. A designer enables a developer to write managed code behind a

10   document and components on them. A Tools/Options dialog provides specific options for various ranges of IDE functionality. A special page can exist for each of the designer hosted applications to enable flexibility for the designer to behave in various ways. The special page includes preferences that the user desires to be invoked when the unmanaged application becomes a designer in the IDE.

15   For document-based designers, there is an additional aspect of customizing application behavior inside the IDE in order to enhance user experience. One example of such functionality is the modified document context menus. Additionally, a "View Code" command is added to some menus to reflect that the document is "designed" and can be programmed against. Still further, a special "designer" toolbar is added for hosted

20   applications to emphasize the designing environment.

Referring now to FIG. 6, there is illustrated a block diagram of a computer operable to execute the disclosed architecture. In order to provide additional context for various aspects of the present invention, FIG. 6 and the following discussion are intended to provide a brief, general description of a suitable computing environment 600 in which

25   the various aspects of the present invention may be implemented. While the invention has been described above in the general context of computer-executable instructions that may run on one or more computers, those skilled in the art will recognize that the invention also may be implemented in combination with other program modules and/or as a combination of hardware and software.

30   Generally, program modules include routines, programs, components, data structures, etc., that perform particular tasks or implement particular abstract data types.

Moreover, those skilled in the art will appreciate that the inventive methods may be practiced with other computer system configurations, including single-processor or multiprocessor computer systems, minicomputers, mainframe computers, as well as personal computers, hand-held computing devices, microprocessor-based or

5   programmable consumer electronics, and the like, each of which may be operatively coupled to one or more associated devices.

The illustrated aspects of the invention may also be practiced in distributed computing environments where certain tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing

10   environment, program modules may be located in both local and remote memory storage devices.

A computer typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by the computer and includes both volatile and nonvolatile media, removable and non-

15   removable media. By way of example, and not limitation, computer readable media can comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media

20   includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital video disk (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer.

25   Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example,

30   and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other

wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

With reference again to FIG. 6, there is illustrated an exemplary environment 600 for implementing various aspects of the invention that includes a computer 602, the computer 602 including a processing unit 604, a system memory 606 and a system bus 608. The system bus 608 couples system components including, but not limited to, the system memory 606 to the processing unit 604. The processing unit 604 may be any of various commercially available processors. Dual microprocessors and other multi-processor architectures may also be employed as the processing unit 604.

The system bus 608 can be any of several types of bus structure that may further interconnect to a memory bus (with or without a memory controller), a peripheral bus, and a local bus using any of a variety of commercially available bus architectures. The system memory 606 includes read only memory (ROM) 610 and random access memory (RAM) 612. A basic input/output system (BIOS) is stored in a non-volatile memory 610 such as ROM, EPROM, EEPROM, which BIOS contains the basic routines that help to transfer information between elements within the computer 602, such as during start-up. The RAM 612 can also include a high-speed RAM such as static RAM for caching data.

The computer 602 further includes an internal hard disk drive (HDD) 614 (*e.g.,* EIDE, SATA), which internal hard disk drive 614 may also be configured for external use in a suitable chassis (not shown), a magnetic floppy disk drive (FDD) 616, (*e.g.,* to read from or write to a removable diskette 618) and an optical disk drive 620, (*e.g.,* reading a CD-ROM disk 622 or, to read from or write to other high capacity optical media such as the DVD). The hard disk drive 614, magnetic disk drive 616 and optical disk drive 620 can be connected to the system bus 608 by a hard disk drive interface 624, a magnetic disk drive interface 626 and an optical drive interface 628, respectively. The interface 624 for external drive implementations includes at least one or both of Universal Serial Bus (USB) and IEEE 1394 interface technologies.

The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions, and so forth. For the computer 602, the drives and media accommodate the storage of any data in a suitable digital format. Although the description of computer-readable media above refers to a

HDD, a removable magnetic diskette, and a removable optical media such as a CD or DVD, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as zip drives, magnetic cassettes, flash memory cards, cartridges, and the like, may also be used in the exemplary operating environment, and

5 further, that any such media may contain computer-executable instructions for performing the methods of the present invention.

A number of program modules can be stored in the drives and RAM 612, including an operating system 630, one or more application programs 632, other program modules 634 and program data 636. All or portions of the operating system, applications,

10 modules, and/or data can also be cached in the RAM 612.

It is appreciated that the present invention can be implemented with various commercially available operating systems or combinations of operating systems.

A user can enter commands and information into the computer 602 through one or more wired/wireless input devices, *e.g.*, a keyboard 638 and a pointing device, such as a

15 mouse 640. Other input devices (not shown) may include a microphone, an IR remote control, a joystick, a game pad, a stylus pen, touch screen, or the like. These and other input devices are often connected to the processing unit 604 through an input device interface 642 that is coupled to the system bus 608, but may be connected by other interfaces, such as a parallel port, an IEEE 1394 serial port, a game port, a USB port, an

20 IR interface, etc.

A monitor 644 or other type of display device is also connected to the system bus 608 *via* an interface, such as a video adapter 646. In addition to the monitor 644, a computer typically includes other peripheral output devices (not shown), such as speakers, printers etc.

25 The computer 602 may operate in a networked environment using logical connections *via* wired and/or wireless communications to one or more remote computers, such as a remote computer(s) 648. The remote computer(s) 648 may be a workstation, a server computer, a router, a personal computer, portable computer, microprocessor-based entertainment appliance, a peer device or other common network node, and typically

30 includes many or all of the elements described relative to the computer 602, although, for purposes of brevity, only a memory storage device 650 is illustrated. The logical

connections depicted include wired/wireless connectivity to a local area network (LAN) 652 and/or larger networks, *e.g.*, a wide area network (WAN) 654. Such LAN and WAN networking environments are commonplace in offices, and companies, and facilitate enterprise-wide computer networks, such as intranets, all of which may connect to a

5   global communication network, *e.g.*, the Internet.

When used in a LAN networking environment, the computer 602 is connected to the local network 652 through a wired and/or wireless communication network interface or adapter 656. The adaptor 656 may facilitate wired or wireless communication to the LAN 652, which may also include a wireless access point disposed thereon for

10   communicating with the wireless adaptor 656. When used in a WAN networking environment, the computer 602 can include a modem 658, or is connected to a communications server on the LAN, or has other means for establishing communications over the WAN 654, such as by way of the Internet. The modem 658, which may be internal or external and a wired or wireless device, is connected to the system bus 608 *via*

15   the serial port interface 642. In a networked environment, program modules depicted relative to the computer 602, or portions thereof, may be stored in the remote memory/storage device 650. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

20   The computer 602 is operable to communicate with any wireless devices or entities operably disposed in wireless communication, *e.g.*, a printer, scanner, desktop and/or portable computer, portable data assistant, communications satellite, any piece of equipment or location associated with a wirelessly detectable tag (*e.g.*, a kiosk, news stand, restroom), and telephone. This includes at least Wi-Fi and Bluetooth™ wireless

25   technologies. Thus, the communication may be a predefined structure as with conventional network or simply an ad hoc communication between at least two devices.

Wi-Fi, or Wireless Fidelity, allows connection to the Internet from a couch at home, a bed in a hotel room or a conference room at work, without wires. Wi-Fi is a wireless technology like a cell phone that enables such devices, *e.g.*, computers, to send

30   and receive data indoors and out; anywhere within the range of a base station. Wi-Fi networks use radio technologies called IEEE 802.11 (a, b, g, etc.) to provide secure,

reliable, fast wireless connectivity. A Wi-Fi network can be used to connect computers to each other, to the Internet, and to wired networks (which use IEEE 802.3 or Ethernet). Wi-Fi networks operate in the unlicensed 2.4 and 5 GHz radio bands, with an 11 Mbps (802.11b) or 54 Mbps (802.11a) data rate or with products that contain both bands (dual band), so the networks can provide real-world performance similar to the basic 10BaseT wired Ethernet networks used in many offices.

Referring now to FIG. 7, there is illustrated a schematic block diagram of an exemplary computing environment 700 in accordance with the present invention. The system 700 includes one or more client(s) 702. The client(s) 702 can be hardware and/or software (e.g., threads, processes, computing devices). The client(s) 702 can house cookie(s) and/or associated contextual information by employing the present invention, for example. The system 700 also includes one or more server(s) 704. The server(s) 704 can also be hardware and/or software (e.g., threads, processes, computing devices). The servers 704 can house threads to perform transformations by employing the present invention, for example. One possible communication between a client 702 and a server 704 may be in the form of a data packet adapted to be transmitted between two or more computer processes. The data packet may include a cookie and/or associated contextual information, for example. The system 700 includes a communication framework 706 (e.g., a global communication network such as the Internet) that can be employed to facilitate communications between the client(s) 702 and the server(s) 704.

Communications may be facilitated via a wired (including optical fiber) and/or wireless technology. The client(s) 702 are operably connected to one or more client data store(s) 708 that can be employed to store information local to the client(s) 702 (e.g., cookie(s) and/or associated contextual information). Similarly, the server(s) 704 are operably connected to one or more server data store(s) 710 that can be employed to store information local to the servers 704.

What has been described above includes examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill in the art may recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all

such alterations, modifications and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term "includes" is used in either the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted when employed as a

5 transitional word in a claim.